Top Gun : The Ultimate Race

Group Members: Claire DiPerna, Andrew Glenn, Divya Tyagi

Project Summary:

The goal of this project was to develop an interactive racing game in the C++ programming language based on the film, Top Gun: Maverick from 2022. Not only is this game engaging, but it also incorporates technical aeronautic themes in attempts of creating an educational and fun experience for the user. In AERSP 424, the team learned about the core features of object-oriented programming, such as encapsulation, inheritance, and polymorphism. A plethora of concepts taught in class were kept in mind when developing this racing game program. The iterative process of design, implementation and verification was better understood by the completion of this game's development—the team learned about the importance of creating organized, well-documented code that passed all test cases and was compatible with different operating systems.

The game commences with a display of the team's logo through the implementation of ASCII art, which is converted to a text file format and read in through the input stream class to operate on files. The logo of the racing game, Top Gun : The Ultimate Race, should populate every time the program is executed.

Next, the game will prompt the user to enter his or her choice for the aircraft pilot based on a selection of four characters: Maverick, Iceman, Rooster and Hangman. The user's pilot string input will be converted to all lowercase characters for ease of recognition, a functionality implemented throughout the entirety of the program. Each pilot has associated statistics assigned through a map's containers, and they are numerical values pertaining to the character's experience, accuracy and handling skills. These stats are displayed to the user on the console as a fraction out of ten.

The user will then be asked to select his or her desired combat aircraft based on the following four models: F-14, F-15ex, F-16, and F-18. Based on the fighter aircraft chosen, initial values for maximum velocity, co-pilot dependency, and maneuverability statistics are matched to the model. Again, this information is displayed in the console for the user's knowledge. If the boolean data type associated with the co-pilot dependency variable is true, the program will run through copilot logic. Essentially, if the fighter aircraft can hold a co-pilot, then the player will be able to select from the following two characters: Goose and Bob.

Each co-pilot has associated statistics detailing their experience, communication and reflex skills, all of which are pre-assigned within the program. Similar to before, these skills are clearly depicted in the console window as a fraction out of ten. For the case where an aircraft model is chosen that has no room for a co-pilot, then a dummy instance is created assigning a value of 0 to all skills associated with said co-pilot.

Finally, the actual game can begin through implementation of racing logic and visual simulation effects. Based on a pre-set distance in miles, the number of iterations needed to accommodate the race length can be determined. The crew and airplane must deal with approaching targets, and they have the ability to shoot projectiles as they deem necessary through pressing the space bar on their keyboards. Speed calculations (both left/right and forward) are done internally based on the model's maximum velocity and pilot's experience and handling skills. If there is a co-pilot, their respective contributions also play a role on the speed the plane can travel. These calculations are handled internally rather than through outputs on the console for added simplicity to the user.

[add more as developed]

Justification:

Course Content:

The following topics were implemented into the racing game program: global & static variables, operators, conditional statements, iterative statements, functions, preprocessors, pointer, function parameter passing, containers, classes, objects, GUI, date and time, file system handling and mouse and keyboard interaction. In addition to using concepts learned in this object-oriented programming course, the developed game also consists of technical knowledge from aeronautics class.

Readability:

One of the team's primary objectives was to ensure the source code was easy to read. Through preprocessors in the form of include directives, the main source code is concise, pulling from various header files to maintain proper structure.

Reusability:

This program is successful in its modularity, breaking up sections into functions and classes based on similar tasks. Having followed typical coding standards also contributes to the program's reusability, as it ensures consistency and ease of readability to any future developers that may look or modify such code. Finally, this code has been tested under various case inputs to ensure it is robust; this increases the likelihood of reusability with no issues.

Usability:

This program should be OS independent, IDE independent and backward and forward compatible, meaning it works regardless of the version of C++ running. Additionally, the logic behind this game is not overly complex, merely requiring user inputs and keyboard interaction for the duration of the race.

Documentation:

This document should be referenced in conjunction with the ReadMe text file before attempting to play the racing game, Top Gun: The Ultimate Race, for added clarity. Depending on the user's interest, there are additional comments throughout the entirety of the source code explaining the implementation of all functions.

Efficiency:

Maximizing efficiency within the program was of utmost importance, specifically through developing code that was reasonable in length and well-handled using functions and classes. In class, the team learned about concepts such as inheritance and polymorphism, where variables could be shared amongst classes and hence limit memory usage.

ReadMe:
(Program developed in Visual Studio 2022 using C++14)

1. Download the included project zip file.
2. Open the project folder in Visual Studio.
3. The game is to be executed from the main.cpp source file, which should open up a separate console window prompting user inputs as the game progresses.
4. Follow the instructions within the game.

Future Work:
The team hopes to create future developments to this program, especially accounting for multi-player functionality where only one user is deemed winner. Additionally, the team wishes to enable flexibility with pilot/co-pilot statistics so that the user is able to boost performances of their chosen character(s).